Journal of Cryptographic Engineering (2021) 11:119–133
https://doi.org/10.1007/s13389-020-00227-6

REGULAR PAPER

Detecting faults in inner product masking scheme

IPM-FD: IPM with fault detection (extended version*)

Wei Cheng[1] · Claude Carlet[2] · Kouassi Goli[1] · Jean-Luc Danger[1,3] · Sylvain Guilley[1,3,4]

Received: 15 November 2019 / Accepted: 28 April 2020 / Published online: 30 May 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract
Side-channel analysis and fault injection attacks are two typical threats to cryptographic implementations, especially in modern embedded devices. Thus, there is an insistent demand for dual side-channel and fault injection protections. As we know, masking is a kind of provable countermeasure against side-channel attacks. Recently, inner product masking (IPM) was proposed as a promising higher-order masking scheme against side-channel analysis, but not for fault injection attacks. In this paper, we devise a new masking scheme named IPM-FD. It is built on IPM, which enables fault detection. This novel masking scheme has three properties: the security orders in the word-level probing model, bit-level probing model and the number of detected faults. IPM-FD is proven secure both in the word-level and in the bit-level probing models and allows for end-to-end fault detection against fault injection attacks. Furthermore, we illustrate its security order by interpreting IPM-FD as a coding problem and then linking it to one defining parameters of linear code and show its implementation cost by applying IPM-FD to AES-128.

Keywords Side-channel analysis · Fault injection attacks · Inner product masking · Fault detection

This work is an extension of [8] (PROOFS 2019).

✉ Wei Cheng
wei.cheng@telecom-paris.fr

Claude Carlet
claude.carlet@univ-paris8.fr

Kouassi Goli
kouassi.goli@polytechnique.edu

Jean-Luc Danger
jean-luc.danger@telecom-paris.fr

Sylvain Guilley
sylvain.guilley@secure-ic.com

1 LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, France

2 LAGA, Department of Mathematics, University of Paris 8, Paris, France

3 Secure-IC S.A.S., Cesson-Sévigné, France

4 Département d’informatique de l’ENS, ENS, CNRS, PSL Research University, Paris, France

1 Introduction

With the advent of Internet of Things (IoT), more and more cryptographic libraries are implemented in software. Now, IoT objects are, most of the time, not made of secure hardware. Therefore, it is important for the software to protect itself in a sound manner. In this article, we assume that the implementation is free from configuration and coding bugs. Still, in this case, attackers can leverage two techniques to extract information: *side-channel* and *fault injection* analyses. Indeed, it is known that a single faulty encryption in AES can fully disclose 128 bits of the secret key [1]. It can be noted that some combined side-channel and fault analyses exist against protected implementations [7,11].

On the one hand, protections against side-channel analysis aim at reducing the signal-to-noise ratio (see definition in [24, § 4.3.2]) an attacker can get. One option is to balance the leakage, a technique which is used to linearize the control flow. For instance, cache-timing attacks can be alleviated by removing conditional opcodes whose condition is sensitive and sensitive pointer dereferencing. Besides, we assume Meltdown and ZombieLoad attack categories are irrelevant as the code we are interested in is at the baremetal level. Still, there is the possibility of sensitive value leakage, which is properly addressed by randomization (*masking* [24, Chap. 9]). Indeed, sensitive values leak through a non-injective and noisy channel; thence, single trace attacks are unpractical.

On the other hand, protections against fault injection attacks boil down to detection of errors, using either spatial, temporal or information redundancy. Other techniques rely on invariant checking, such as idempotence of encryption composed by decryption.

In this paper, we present a joint countermeasure to both attacks, which is more efficient than two countermeasures piled one on top of each other.

State of the art. In scientific literature, early countermeasures against both side-channel and fault injection attacks have been designed in hardware. Several gate-level logic styles have been introduced, in particular dual rail with precharge logic, aiming at balancing the leakage. Namely, redundant encodings, where each bit a is represented as a pair of bits (a_f, a_t), such that $a_f = \neg a_t = a$ during computation evaluation phase. Owing to this redundancy, the total number of bits set to 1 is unchanged (if in addition, the *evaluation* phase is interleaved with a *precharge* phase, the Hamming distance between two states is also constant, irrespective of the sensitive data manipulated). Besides, the redundant encoding $a_f = \neg a_t = a$ allows for computation checks, as in evaluation phase, $a_f = a_t$ (two configurations, namely $(0, 0)$ and $(1, 1)$, are forbidden). Starting from wave dynamic differential logic (WDDL [24, Chap. 7]), other improvements have been successively introduced (MDLP, iMDPL [21], ParTI [33], etc.) Also, some exotic styles have been proposed (asynchronous logic [27], adiabatic logic [26], etc.). All this corpus requires hardware support.

In this paper, we target software-level countermeasures. We build upon the higher-order side-channel countermeasure known as IPM [2] to enrich it to detect faults injected during the computation.

Contributions. We devise an end-to-end fault-detection scheme which operates from within a provable high-order multivariate masking scheme. In practice, we enhance IPM scheme to enable end-to-end side-channel and fault injection detection, while keeping security proofs in the probing security model. Furthermore, we quantify the impact of both side-channel and fault detection on a complete AES-128 to show the advantages of our new scheme.

This work is an extension of the previous eponymous conference paper [8]. We highlight below the new extensions incorporated in this paper:

- The generalization of IPM and IPM-FD to (O)DSM is presented to emphasize the connections and differences between two schemes. This generalization allows us to optimize the former by using constructions of the latter in a coding theoretic approach. For instance, some optimal codes in (O)DSM would also be applicable in IPM and IPM-FD.
- We clarify the fault models by showing the essential different assumptions under these models, which determine the fault detection capability of IPM-FD and (O)DSM. We insist that our IPM-FD only considers the last two fault models since we focus on the end-to-end protections.
- By comparing the IPM-FD and BM-FD (Boolean masking with fault detection), we demonstrate the advantages of the former over the latter. Specifically, IPM-FD needs less shares to achieve the same security order at word level. Furthermore, the bit-level security order of IPM-FD can be much higher than BM-FD given the same number of shares.
- We insist that the systematic construction of optimal codes for IPM-FD and DSM at both word level and bit level is still an open problem. In this paper, we only provide the metrics and some results with small number of shares by an exhaustive study. Note that another exhaustive study for optimal linear codes for IPM is also available in a related specialized paper [10].

Outline. The rest of this paper is organized as follows: Sect. 2 introduces two typical schemes as the state of the art of countermeasures. Our novel protection is presented in Sect. 3, with security analysis and optimal code selection in Sect. 4. The practical performance evaluation is presented in Sect. 5. Finally, Sect. 6 concludes the paper and opens some perspectives.

2 State of the art on side-channel and fault protection

Side-channel protections considered in this work come in two flavors:

- Inner product masking (IPM) [2] is a word-oriented (e.g., byte-oriented) masking scheme, equipped with universal operations (namely addition and multiplication). It is optimized to resist attacks at both the word-level and bit-level probing model [30], which is suitable for computing cryptographic algorithms that are subject to high-order side-channel analysis.
- Direct sum masking (DSM) [5] is a masking scheme which allows for concurrent side-channel and fault injection protection. It expresses the masking as the two encodings of the secret in a code C, and masks in a code D, respectively. This allows us to recover the information by decoding from C and to check the masks by decoding from D.

These two protections are presented, one after the other, in this section.

2.1 Inner product masking

2.1.1 Notations

Computations are carried out in characteristic two finite fields: $\mathbb{F}_2$ for bits and $\mathbb{K}$ for larger fields. In practice $\mathbb{K}$ can be $\mathbb{F}_{2^l}$ for some l, e.g., $l = 8$ for AES, and $l = 4$ for PRESENT. The elements from $\mathbb{K}$ are termed *words*, and they are also referred to as *bytes* when $l = 8$ and to *nibbles* when $l = 4$. We denote $+$ the addition in characteristic two fields $\mathbb{K}$, which is bitwise XOR. Recall that the subtraction is the same operation as the addition in $\mathbb{K}$. Elements of $\mathbb{F}_2$ are denoted as $\{0, 1\}$, and elements of $\mathbb{F}_{2^l}$ (as *words*) are represented as polynomials. In this paper, we use $\mathbb{F}_{2^4} \cong \mathbb{F}_2[\alpha]/\langle\alpha^4 + \alpha + 1\rangle$, and $\mathbb{F}_{2^8} \cong \mathbb{F}_2[\alpha]/\langle\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1\rangle$ (that of AES).

We recall that linear codes are space vectors, characterized by their base field $\mathbb{K}$, their length n and their dimension k. In addition, linear codes have parameters traditionally denoted as $[n, k, d]$, where d is the minimum distance. The dual of a linear code D is the linear code $D^\perp$ whose code words are orthogonal to all code words of D. The dual distance $d^\perp$ of a linear code D happens to be equal to the minimum distance of $D^\perp$ [23].

Let n be the number of shares in IPM, and the coefficient vector in IPM is $\mathbf{L} = (L_1, L_2, \ldots, L_n)$ where $L_1 = 1$ for performance reason [2, § 1.2].

Definition 1 *(IPM data representation)* A word of secret information $X \in \mathbb{K}$ is represented in IPM as a tuple of n field elements:

$$\mathbf{Z} = \left(X + \sum_{i=2}^{n} L_i M_i, M_2, \ldots, M_n\right) = X\mathbf{G} + \mathbf{MH} \qquad (1)$$

where $\mathbf{M} = (M_2, M_3, \ldots, M_n)$ is the mask materials, and $\mathbf{G}$ and $\mathbf{H}$ are generating matrices of linear codes C and D, respectively, as shown below:

$$\mathbf{G} = \begin{pmatrix} 1 \mid 0 \ 0 \ \ldots \ 0 \end{pmatrix} \in \mathbb{K}^{1\times n}, \qquad (2)$$

$$\mathbf{H} = \begin{pmatrix} L_2 & \mid & 1 & 0 & \ldots & 0 \\ L_3 & \mid & 0 & 1 & \ldots & 0 \\ \vdots & \mid & 0 & 0 & \ddots & 0 \\ L_n & \mid & 0 & 0 & \ldots & 1 \end{pmatrix} \in \mathbb{K}^{(n-1)\times n}. \qquad (3)$$

The secret information X can be demasked by inner product between two vectors as: $X = \langle\mathbf{L}, \mathbf{Z}\rangle = \sum_{i=1}^{n} L_i Z_i$. Finally, we introduce some handy subset notations. Let $\mathbf{Z} = (Z_1, \ldots, Z_n) = (Z_i)_{i\in\{1,\ldots,n\}}$ be a vector. We have:

$$\mathbf{Z}_I = (Z_i)_{i\in I} \quad \text{for} \quad I \subseteq \{1, \ldots, n\}.$$

For instance, $Z_{\{i\}\cup\{k+1,\ldots,n\}}$, for $1 \le i \le k \le n$, represents the $(n - k + 1)$ vector $(Z_i, Z_{k+1}, Z_{k+2}, \ldots, Z_n)$.

2.1.2 Security order regarding side-channel analysis

The security of IPM is stated in the *probing model* [17]: The security order is the maximum number of shares which are independent to masked information. We clarify word-level and bit-level security orders as follows:

- **Word-level (l-bit) security order d_w**: Since many devices perform computation on word-level data, byte-level operations are very common especially on embedded devices. In this paper, we also present instances for 4-bit (nibble) variables for adopting IPM to protect implementation of lightweight cipher like PRESENT, Simon, Speck, etc.
- **Bit-level security order d_b**: In practice, each bit of sensitive variable can be investigated independently and/or several bits can be evaluated jointly. We consider here the number of bits that can be probed by attackers in one time, which is consistent with the bit-level probing model proposed by Poussier *et al.* [30].

The main advantage of IPM is the higher bit-level security order than Boolean masking, which is called "security order amplification" in [36]. It has been proven in [30] that side-channel resistance is directly connected to the dual distance $d_D^\perp$ of the code D generated by $\mathbf{H}$. Precisely, the security order t of IPM is equal to $t = d_D^\perp - 1$ [30].

The dual distance of linear code D is equal to the minimum distance of the dual code $D^\perp$ [23]. It is easy to see that the latter has dimension 1 and is generated by a $1 \times n$ matrix:

$$\mathbf{H}^\perp = \begin{pmatrix} 1 & L_2 & L_3 & \ldots & L_n \end{pmatrix}. \qquad (4)$$

In order to investigate the bit-level security, the definition of expansion is introduced as follows.

Definition 2 *(Code Expansion)* By using subfield representation, the elements in $\mathbb{K} = \mathbb{F}_{2^l}$ are decomposed into $\mathbb{F}_2$, and we have:

$$\begin{aligned} &\text{SubfieldRepresentation:} \\ &\quad (1, L_2, \ldots, L_n)_{2^l} \longrightarrow (I_l, \mathbb{L}_2, \ldots, \mathbb{L}_n)_2, \end{aligned} \qquad (5)$$

where I_l is the $l \times l$ identity matrix in $\mathbb{F}_2$ and $\mathbb{L}_i$ $(2 \le i \le n)$ are $l \times l$ matrices.

To derive the matrices, we can use that $\mathbb{F}_{2^l}$ is a field extension of $\mathbb{F}_2$, and given an irreducible polynomial P over $\mathbb{F}_2$ and denoting each element $a \in \mathbb{F}_{2^l}$ as $\sum_{i=0}^{l-1} a_i\alpha^i$ [mod $P(\alpha)$], replace a by $(a_0, \ldots, a_{l-1})$. Under the computer algebra system `Magma` [35], P is `DefiningPolynomial(`$\mathbb{F}_{2^l}$`)` and

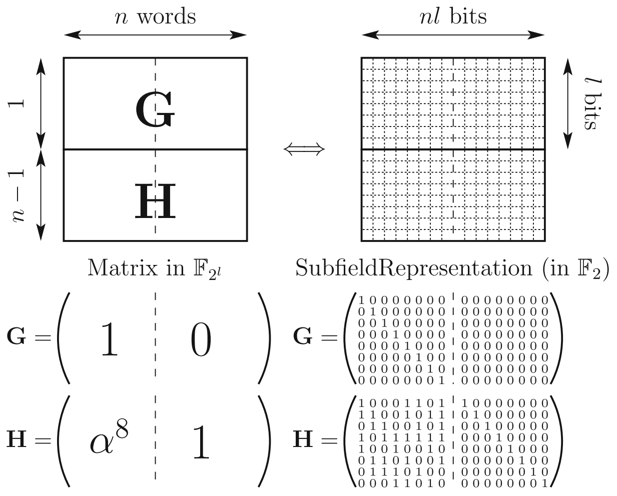

Fig. 1 Dimensions of (typical) IPM encodings, for $n = 2$, on $l = 8$ bits at byte level

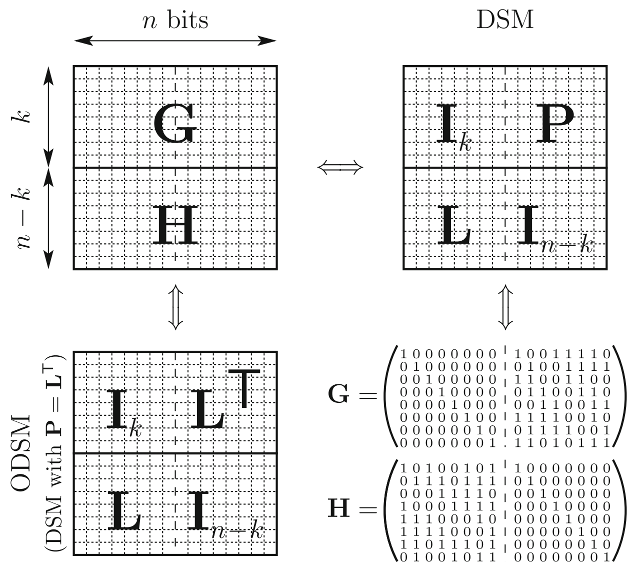

Fig. 2 Dimensions of (typical) DSM and ODSM encodings (on $\mathbb{F}_2$), for $k = 8$ bit and $n = 16$ bit

D' is the representation of D in subfield (`SubfieldRepresentationCode(`D`)`). If D has parameters $[n, k, d]_{2^l}$, then D' has parameters $[nl, kl, d']_2$, where $d' \geq d$. IPM opportunistically exploits the fact that this inequality can be strict, and attempts to maximize the difference $d' - d$.

At word level, we notice that the dual distance of D is equal to n as long as $\forall i,\ L_i \neq 0$. As a result, the word-level security order of IPM is $d_w = n - 1$ which is in consistence with [2]. In addition, security order d_b at the bit level of IPM is equal to the dual distance of the code expanded by D from $\mathbb{F}_{2^l}$ to $\mathbb{F}_2$. A typical example of IPM codes matrices $\mathbf{G} = (1, 0)$ and $\mathbf{H} = (L_2 = \alpha^8, 1)$ in $\mathbb{F}_{2^8}$ is given in Fig. 1. The security order at word (byte) level is $d_w = n-1 = 1$ and at bit level is $d_b = 3$ because the dual code of $D = \text{span}(H)$ is generated by $(1, L_2)$, which, after projection in $\mathbb{F}_2$, has parameters $[16, 8, 4]_2$.

Moreover, addition and multiplication are proven to be $t = (n - 1)$-order secure at word level in [3] using t-SNI property [4]; thus, the word-level security order is maintained by composition. Still, when a variable is reused, caution must be taken where a refresh algorithm is always adopted to avoid dependence. The refresh operation allows us to decorrelate two copies of a variable that need to be used at two places (to avoid side-channel flaws as put forward in [14]). However, IPM cannot detect faults since no redundancy is inserted to the coding.

2.2 Direct sum masking

Direct sum masking has been originally introduced as orthogonal direct sum masking (ODSM [5]). The secret $\mathbf{X}$ is represented as a bit vector in $\mathbb{F}_2^l$. It is encoded using generating matrix $\mathbf{G}$ (of size $l \times nl$ in $\mathbb{F}_2$) as a word in $\mathbb{F}_2^{nl}$. Some random masks $\mathbf{M}$, drawn uniformly in $\mathbb{F}_2^{(n-1)l}$, are encoded with matrix $\mathbf{H}$ (of size $(n-1)l \times nl$). After masking the secret with the mask materials, one gets the protected information:

$$\mathbf{Z} = \mathbf{XG} + \mathbf{MH}. \tag{6}$$

The features of the DSM are the following:

- Elements are bits;
- Computation on masked variable $\mathbf{Z}$ occurs matricially;
- Side-channel protection is ensured at order $d_D^\perp - 1$;
- Fault detection allows detecting $d_C - 1$ bitflips.

Orthogonal direct sum masking (ODSM) is a particular case of DSM for which $\mathbf{GH}^\mathsf{T} = 0_{k\times(n-k)}$, or said differently, C and D are mutually dual codes. An illustration of DSM and ODSM is provided in Fig. 2. In this figure, without loss of generality, the matrices $\mathbf{G}$ and $\mathbf{H}$ are written in systemic form. The conditions for $C = \text{span}(\mathbf{G})$ and $D = \text{span}(\mathbf{H})$ to be complementary are recalled in the following:

Lemma 1 ([28, Proposition 1]) *Let* $0 \leq k \leq n$, *and*

$$\mathbf{G} = \begin{pmatrix}\mathbf{I}_k & \mathbf{P}\end{pmatrix} \in \mathbb{F}_2^{k\times n} \text{ and } \mathbf{H} = \begin{pmatrix}\mathbf{L} & \mathbf{I}_{n-k}\end{pmatrix} \in \mathbb{F}_2^{(n-k)\times n}.$$

Then, the following three statements are equivalent:

1. $\begin{pmatrix}\mathbf{G}\\ \mathbf{H}\end{pmatrix} \in \mathbb{F}_2^{n\times n}$ *is invertible;*
2. $I_k + \mathbf{PL}^\mathsf{T} \in \mathbb{F}_2^{k\times k}$ *is invertible;*
3. $I_{n-k} + \mathbf{LP}^\mathsf{T} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ *is invertible.*

A detailed comparison between DSM and IPM is proposed in Table 1.

Table 1 Comparison between (O)DSM and IPM-FD schemes

Features	(O)DSM [5]	IPM [2]	Comments
Objects	Bits	Words	IPM can always be seen as a DSM scheme by subfield representation. Reverse compatibility only if bit vectors matrix multiplication can be promoted in $\mathbb{F}_{2^l}$
Operations	Matrix product	Adapted Ishai-Sahai -Wagner (ISW) [17]	ISW has been studied extensively
Side-channel protection	$d_D^\perp - 1$ is the protection order	Same, albeit with two notions: word and bit levels	For real-world (power/electromagnetic) attacks, bit-level security is relevant [15]
Fault injection protection	$d_C - 1$ bitflips are detected	IPM-FD: repetition code (this paper)	IPM-FD could be empowered by using a better or even optimal code instead of repetition code

$$\begin{tikzcd}
\vec{X} \arrow[r, shift left, "E_C"] & \vec{X}\mathbf{G} \arrow[l, shift left, "D_C"] \arrow[dr] & & \\
& & + \arrow[r] & \vec{Z} = \vec{X}\mathbf{G} + \vec{M}\mathbf{H} \arrow[ull, "\Pi_C"'] \arrow[dll, "\Pi_D"] \\
\vec{M} \arrow[r, shift left, "E_D"] & \vec{M}\mathbf{H} \arrow[l, shift left, "D_D"] \arrow[ur] & &
\end{tikzcd}$$

Fig. 3 Commutative diagram of DSM masking scheme with encoding and decoding

On the contrary to IPM, the matrices **G** and **H** do not have specific form (recall IPM matrices are formatted as Eqn. 2 and Eqn. 3). However, there is no general inverse operation of "SubfieldRepresentation" (recall Def. 2) for DSM. Therefore, IPM is a special case of DSM, but some DSM encodings (Eqn. 6) cannot be represented as IPM.

ODSM uses orthogonal codes such that recovering **M** is straightforward knowing **Z**: It consists in an orthogonal projection from space vector $\mathbb{F}_2^{nl}$ onto D. Actually, the complete commutative diagram involved in DSM is depicted in Fig. 3. The operations are explicited below:

- Information vector **X** is encoded as **XG** (using linear application E_C), while decoding of **XG** into **X** is ensured by the decoding application D_C;
- Similarly, masking random variables **M** are encoded as **MH** (using linear application E_D). Decoding of **MH** into **M** is ensured by the decoding application D_D;
- Creating an encoded word **Z** consists of adding one code word **XG** from C to one code word **MH** from D. In reverse, projections of $\mathbf{Z} \in \mathbb{F}_2^{nl}$ to C (resp. D) are obtained by linear projection operation Π_C (resp. Π_D).

When C and D are orthogonal, then $\mathbf{G}\mathbf{H}^\mathsf{T} = \mathbf{0}$, the all-zero $l \times (n-1)l$ matrix. As a result, we have $\Pi_C(\mathbf{Z}) = \mathbf{Z}\mathbf{G}^\mathsf{T}(\mathbf{G}\mathbf{G}^\mathsf{T})^{-1}\mathbf{G}$ and $\Pi_D(\mathbf{Z}) = \mathbf{Z}\mathbf{H}^\mathsf{T}(\mathbf{H}\mathbf{H}^\mathsf{T})^{-1}\mathbf{H}$ as in [5].

This allows for the verification that an attacker who injects a fault has not corrupted (useless in terms of exploitation) the masks **M**. In practice, the attack (addition of a nonzero bit vector $\epsilon \in \mathbb{F}_2^{nl}\backslash\{0\}$) is undetected if and only if $\epsilon \in C$. Indeed, otherwise ϵ has a nonzero component in space vector D, and the fault injection is detected. The fault detection capability can be quantified in two models:

1. Assumption 1: The difficulty of the attack is larger if the number of flipped bits is larger. Thus, undetected faults $\epsilon \in C\backslash\{0\}$ must have Hamming weights $\geq d_C$, where d_C is the minimum distance of code C.
2. Assumption 2: The attacker can corrupt Z regardless of the value of ϵ, but cannot control the value of ϵ. Said differently, ϵ is a random variable uniformly distributed in $\mathbb{F}_2^{nl}\backslash\{0\}$. This fault is undetected provided $\epsilon \in C\backslash\{0\}$. As C has dimension l, the cardinality of $C\backslash\{0\}$ is $2^l - 1$. Therefore, the probability that the fault is not detected equals $\frac{2^l-1}{2^{nl}-1} \approx 2^{-l(n-1)}$. This number is independent from the code C, but depends on code D.

Thus, the probability of undetected faults gets lower as l and n increase. However, this approach has three drawbacks:

- First of all, the masks used in ODSM remain unchanged during each call of cipher, which allows fault detection. But the "static" masks may pose a vulnerability since masks should be refreshed to avoid unintended dependencies between sensitive variables.

- Secondly, it allows only to check errors on states $\mathbf{Z}$, but not during nonlinear computations (which are tabulated, i.e., operations on $\mathbf{Z}$ consist in lookup table accesses). From a hardware point of view, this means that ODSM allows us to detect faults in sequential logic (e.g., register banks, RAM, etc.), but not in combinational logic (e.g., logic gates or ROM).
- Thirdly, during verification, that is, the projection of $\mathbf{Z}+\epsilon$ in space vector D, the state $\mathbf{Z}$ is manipulated; hence, additional leakage is produced, which must be taken into account in the security evaluation of ODSM representation (Eqn. 6). This is the reason we suggest detecting faults at the very end (end-to-end fault detection), like after encryption or decryption.

The first two points are structural weaknesses and will be fixed in Algorithm 1, starting from Sect. 3. For the third one, some codes suitable for DSM are constructed by Carlet *et al.* in [6] by duplicating the masks $\mathbf{M}$, while this solution does not allow an end-to-end scheme.

3 Novel end-to-end fault detection scheme

3.1 Rationale

The core idea in our new scheme is to duplicate (two or more times) the secret X, rather than duplicating masks $\mathbf{M}$ as in [6], so that it can be checked at the end (when it is no longer sensitive–e.g., a ciphertext is a non-sensitive variable, so as the plaintext).

Our new scheme is a IPM-like masking scheme, called IPM-FD. Since IPM is a promising high-order masking scheme, we extend it with fault detection capability so that it can resist both side-channel analysis and fault injection attacks simultaneously. Specifically, we represent the information as a vector $(X_1, X_2, \ldots, X_k) \in \mathbb{K}^k$ where $\mathbb{K} = \mathbb{F}_{2^l}$.

We propose the new encoding as follows. Let us denote:

Definition 3 *(IPM-FD data representation)* Let $X_i \in \mathbb{K}$ be the k copies of secret information; then, the encoding is represented as a tuple of n elements in $\mathbb{K}$:

$$\mathbf{Z} = (\underbrace{X_1, X_2, \ldots, X_k}_{\text{secrets}}\ \mathbf{X})\ \mathbf{G} + (\underbrace{M_{k+1}, \ldots, M_n}_{\text{masks}}\ \mathbf{M})\ \mathbf{H}$$
$$= (Z_1, Z_2, \ldots, Z_n), \quad (7)$$

where

$$\mathbf{G} = (\ I_k || \ 0) \in \mathbb{K}^{k\times n},$$
$$\mathbf{H} = (\ L\ ||\ I_{n-k}) \in \mathbb{K}^{(n-k)\times n},$$

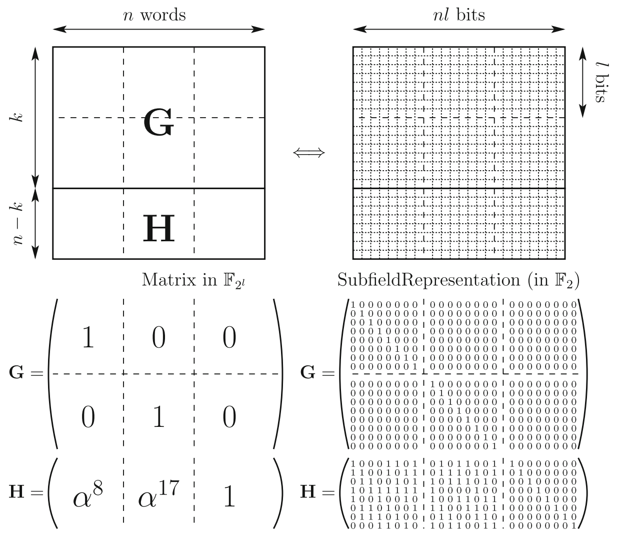

Fig. 4 Dimensions (typical) of IPM-FD encodings, for $n = 3$, $k = 2$ and $l = 8$ bits

where I_k is the $k \times k$ identity matrix in $\mathbb{K}$, and L is a matrix of size $(n-k) \times k$, that is,
L has coefficients $(L_{i,j})_{k<i\le n, 1\le j\le k}$.

This definition 3 is a generalization of Def. 1. In practice, we will call Eqn. 7 with redundancy to detect faults in the information X, i.e., $(X_1, X_2, \ldots, X_k) = (X, X, \ldots, X)$, as:

$$\mathbf{Z} = (X, X, \ldots, X)\mathbf{G} + (M_{k+1}, \ldots, M_n)\mathbf{H}. \quad (8)$$

For the sake of convenience, the IPM-FD encoding used in this paper is depicted in Fig. 4. It illustrates a protection using $n = 3$ shares of $l = 8$ bits, with the following security features:

- $d_w = 1$ (first-order secure at byte level), because dual distance of $\mathbf{H}$ in $\mathbb{F}_{2^8}$ is 2;
- $d_b = 3$ (third-order secure at bit level), since the dual distance of the optimal $\mathbf{H}$ over $\mathbb{F}_2$ is 4—the subfield representation (by Def. 2) of the dual code $\mathbf{H}^\perp$ spawn by $\begin{pmatrix}1 & L_2 & L_3\end{pmatrix}$ has parameters $[24, 8, 4]_2$ where we take $L_2 = \alpha^8$ and $L_3 = \alpha^{17}$ as optimal parameters (from an exhaustive search over all possible candidates of L_2 and L_3 over $\mathbb{F}_{2^8}$) in this case (Fig. 4).

Computation can be carried out on such $\mathbf{Z}$, and when it is over (e.g., the complete AES is finished), the implementation can check whether the k copies of the information are the same. This allows us to detect up to $(k-1)$ errors (there is an error if the k copies are not equal to each other). It is worth noting that this model is stronger than the one in ODSM where only errors ϵ with Hamming weight $w_H(\epsilon) > d_C$ are detected in ODSM.

Repeating X k times may increase the signal captured by the attacker by a factor k; however, it is irrelevant to security order. Indeed, there is more signal, but it is correlated; therefore, it has no impact on the amount of information. Notice that, as a future extension, one might consider an encoding of information X which is more efficient in terms of rate than the simple k-times repetition code $X \mapsto (X, \ldots, X)$. However, such representation in Eqn. 8 allows for an end-to-end security protection against fault injection attacks, as illustrated in Algorithm 1.

For fault detection, either Algorithm 1 is started from scratch, or other actions, such as event logging for subsequent analysis (aiming at taking proactive actions to plug this leak), are triggered off. It is obvious that detecting fault in each intermediate phase can be carried out at any place in Algorithm 1, especially during step 5. However, such precaution is superfluous, as an overall check is done at the end, that is, at line 8. In addition, intermediate checks would disclose when the fault occurs (e.g., at which round), which delivers precious feedback to the attacker regarding the accuracy and the reproducibility of the setups.

Algorithm 1: End-to-end protection of a cryptographic algorithm (here AES-128) against fault injection attacks using IPM-FD scheme

```
input : Plaintext $X \in \mathbb{F}_{2^8}^{16}$, key $K \in \mathbb{F}_{2^8}^{16}$, and number of detected
        faults $d_f = k - 1$, number of shares $n = d_w + 1$,
        bit-level security order $d_b = d_D^{\perp} - 1$
output: Ciphertext, or $\perp$ if a fault has been detected
1  The matrices G and H (corresponding to code $C$ and $D$,
   respectively) are determined with respect to the requirements on
   side-channel and fault protection $d_w$, $d_b$ and $d_f$
2  $\mathbf{M} \leftarrow_{\mathcal{R}} \mathbb{F}_{2^8}^{16\times(n-k)}$
3  $\mathbf{Z} \leftarrow (X, \ldots, X)\mathbf{G} + \mathbf{MH}$            // Recall Eqn. 8
4  ...
5  Arithmetic operations for the (secure) computation, using
   Lagrange interpolation polynomial. This includes additions
   (Algorithm 2) and multiplications (Algorithm 4)
6  ...
7  $(X_1, \ldots, X_k) \leftarrow \Pi_C(\mathbf{Z})$    // Recall $\Pi_C(\mathbf{Z})$ in Fig. 3
8  if $X_1 = \ldots = X_k$ then
9      return $X_1$
10 else
11     return $\perp$
```

Therefore, the design of IPM-FD scheme for a specific cryptographic algorithm can be simplified to select good parameters **G** and **H**, which is corresponding to choose good codes for IPM-FD. We first show how to perform basic operations in the next subsection.

3.2 Computing with representation of IPM-FD

First of all, we present one instance of IPM-FD with $k = 2$ to clarify its encoding. We denote $\mathbf{X} = (X_1, X_2) \in \mathbb{K}^2$, and $\mathbf{M} = (M_3, \ldots, M_n) \in \mathbb{K}^{n-2}$. Thus, we have Eqn. 7 such that,

$$\mathbf{G} = \left(\begin{array}{cc|cccc} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \end{array}\right),$$

$$\mathbf{H} = \left(\begin{array}{cc|cccc} L_{3,1} & L_{3,2} & 1 & 0 & \ldots & 0 \\ L_{4,1} & L_{4,2} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & 0 & 0 & \ddots & 0 \\ L_{n,1} & L_{n,2} & 0 & 0 & \ldots & 1 \end{array}\right),$$

or said differently, we have $\mathbf{Z} = (Z_1, \ldots, Z_n) \in \mathbb{K}^n$ which is equal to:

$$Z_1 = X_1 + L_{3,1}M_3 + L_{4,1}M_4 + \ldots + L_{n,1}M_n$$
$$Z_2 = X_2 + L_{3,2}M_3 + L_{4,2}M_4 + \ldots + L_{n,2}M_n$$
$$Z_i = M_i \quad \text{for } 3 \le i \le n$$

Here, we can see that $(Z_1, Z_3, \ldots, Z_n) \in \mathbb{K}^{n-1}$ and $(Z_2, Z_3, \ldots, Z_n) \in \mathbb{K}^{n-1}$ are two IPM sharings [2]. Therefore, we have $k = 2$ ways to demask:

$$\langle L_1, \mathbf{Z}\rangle = X_1 = X, \quad \text{and} \quad \langle L_2, \mathbf{Z}\rangle = X_2 = X,$$

where as a convention, $L_{1,1} = L_{2,2} = 1$, $L_{1,2} = L_{2,1} = 0$ and

$$L_1 = \left(L_{i,1}\right)_{1\le i\le n} \in \mathbb{K}^n, \quad \text{and} \quad L_2 = \left(L_{i,2}\right)_{1\le i\le n} \in \mathbb{K}^n.$$

It is known that universal computation can be achieved by Lagrange interpolation, which only requires addition and multiplication. Hereafter, we present three basic algorithms, with the most general case (k words of information and scalable with different k) used to build a complete masked cryptographic implementation.

3.2.1 Secure addition of IPM-FD

With Eqn. 8, we denote encoding of X and X' by **Z** and **Z**$'$, respectively; thus, the addition is linear and can be calculated straightforwardly as in Algorithm 2.

3.2.2 Secure refresh algorithm for IPM-FD

As suggested in [31], we need to apply a refresh algorithm after each squaring operation to keep independence between masks (Algorithm 4 with $\mathbf{Z} = \mathbf{Z}'$). The algorithm for the refresh of IPM-FD is given in Algorithm 3. Notice that this

Algorithm 2: Secure addition in IPM-FD

```
input : Two sets of scalar tuples $\mathbf{X} = (X_1, \dots, X_k)$ and
        $\mathbf{X'} = (X'_1, \dots, X'_k)$ shared as:
  – $\mathbf{Z} = (Z_1, \dots, Z_n) = (X_1 + \sum_{i=k+1}^{n} L_{i,1} M_i, \dots, X_k + \sum_{i=k+1}^{n} L_{i,k} M_i, M_{k+1}, \dots, M_n) \in \mathbb{K}^n$,
  – $\mathbf{Z'} = (Z'_1, \dots, Z'_n) = (X'_1 + \sum_{i=k+1}^{n} L_{i,1} M'_i, \dots, X'_k + \sum_{i=k+1}^{n} L_{i,k} M'_i, M'_{k+1}, \dots, M'_n) \in \mathbb{K}^n$.
output: A sharing $\mathbf{R} = (R_1, \dots, R_n) \in \mathbb{K}^n$ such that, for all $j$
        $(1 \le j \le k)$,
        $\langle \mathbf{R}_{\{j\} \cup \{k+1,\dots,n\}}, \mathbf{L}_{\{j\} \cup \{k+1,\dots,n\},j} \rangle = X_j + X'_j$
1 $\mathbf{R} = (Z_1 + Z'_1, \dots, Z_n + Z'_n)$
2 return $R$
```

algorithm can be computed in place, meaning that the output overwrites the input.

Algorithm 3: IPM-FD refresh algorithm

```
input : Let $k < n$. One IPM-FD sharing
        $\mathbf{Z} = (X_1, \dots, X_k)\,\mathbf{G} + (M_{k+1}, \dots, M_n)\,\mathbf{H}$, as defined
        in Eqn. 7
output: An equivalent IPM-FD sharing
        $\mathbf{Z}' = (X_1, \dots, X_k)\,\mathbf{G} + (M'_{k+1}, \dots, M'_n)\,\mathbf{H}$, where
        $(M_{k+1}, \dots, M_n)$ is independent from $(M'_{k+1}, \dots, M'_n)$.
1 $\mathbf{Z}' \leftarrow \mathbf{Z}$   // When computed in-place, $\mathbf{Z}'$ is not needed.
2 for $i \in \{k+1, \dots, n\}$ do
3     $\epsilon \leftarrow_{\mathcal{R}} \mathbb{K}$                // Fresh random variable
4     $Z'_i \leftarrow Z'_i + \epsilon$
5     for $j \in \{1, \dots, k\}$ do
6         $Z'_j \leftarrow Z'_j + L_{i,j}\epsilon$
7 return $\mathbf{Z}' \in \mathbb{K}^n$.
```

3.2.3 Secure multiplication of IPM-FD

Secure multiplication can be achieved by selecting only one among the k first coordinates, while keeping the $(n-k)$ masks, and multiplying $(n-k+1)$ shares by using the original IPM multiplication. Therefore, multiplication of IPM-FD is implemented in Algorithm 4.

Multiplication is repeated k times on shares in $\mathbb{K}^{n-k+1}$, and the resulting $\mathbf{P}[j] \in \mathbb{K}^{n-k+1}$ for $j \in \{1, \dots, k\}$ are applied from line 4 to line 6 as in Algorithm 4 to homogenize masks in $(k-1)$ sharings with the same masks as $\mathbf{P}[1]$.

We refer to line 4 to line 6 of Algorithm 4 as the homogenization algorithm used to merge the results $\mathbf{P}[j]$ where $1 \le j \le k$. Thus, we have the following lemma, which applies to non-redundant sharings such as that of Eqn. 1.

Lemma 2 (Homogenization of two sharings) *Let $\mathbf{Z} = (Z_1, \dots, Z_n)$ and $\mathbf{Z}' = (Z'_1, \dots, Z'_n)$ be two sharings that $\langle L, \mathbf{Z} \rangle = X$ and $\langle L', \mathbf{Z}' \rangle = X'$. There exists an equivalent sharing $\mathbf{Z}''$ and an algorithm to transform $\mathbf{Z}'$ into $\mathbf{Z}''$ such that $\mathbf{Z}$ and $\mathbf{Z}''$ share all coordinates but the first one.*

Algorithm 4: Secure multiplication of IPM-FD with k pieces of information

```
input : Two sets of scalar tuples $\mathbf{X} = (X_1, \dots, X_k)$ and
        $\mathbf{X'} = (X'_1, \dots, X'_k)$ shared as:
  – $\mathbf{Z} = (Z_1, \dots, Z_n) = (X_1 + \sum_{i=k+1}^{n} L_{i,1} M_i, \dots, X_k + \sum_{i=k+1}^{n} L_{i,k} M_i, M_{k+1}, \dots, M_n) \in \mathbb{K}^n$,
  – $\mathbf{Z'} = (Z'_1, \dots, Z'_n) = (X'_1 + \sum_{i=k+1}^{n} L_{i,1} M'_i, \dots, X'_k + \sum_{i=k+1}^{n} L_{i,k} M'_i, M'_{k+1}, \dots, M'_n) \in \mathbb{K}^n$.
output: A sharing $\mathbf{P} = (P_1, \dots, P_n) \in \mathbb{K}^n$ such that, for all $j$
        $(1 \le j \le k)$, $\langle \mathbf{P}_{\{j\} \cup \{k+1,\dots,n\}}, \mathbf{L}_{\{j\} \cup \{k+1,\dots,n\},j} \rangle = X_j \cdot X'_j$
1 for $j \in \{1, \dots, k\}$ do
2     $\mathbf{P}[j] \leftarrow \mathsf{IPMult}(Z_{\{j\} \cup \{k+1,\dots,n\}}, Z'_{\{j\} \cup \{k+1,\dots,n\}})$   // IPMult is Algorithm 5 of [2]
3     Let us write $\mathbf{P}[j]$ as $(P_j, N_{k+1,j}, \dots, N_{n,j})$, where
      $P_j = X_j X'_j + \sum_{i=k+1}^{n} L_{i,j} N_{i,j} \in \mathbb{K}$
4 for $j \in \{2, \dots, k\}$ do        // Masks homogenization between $\mathbf{P}[1]$ and $\mathbf{P}[j]$
5     for $i \in \{k+1, \dots, n\}$ do
6         $P_j \leftarrow P_j + L_{i,j}(N_{i,1} + N_{i,j})$
      // $(P_j, N_{k+1,1}, \dots, N_{n,1})$ is a sharing of $X_j X'_j$ by $(n-k)$ masks of $\mathbf{P}[1]$
7 return $\mathbf{P} = \big(P_1, \dots, P_k, N_{k+1,1}, \dots, N_{n,1}\big) \in \mathbb{K}^n$.
```

Proof We apply a pivot technique to $\mathbf{Z}''$. Let $\epsilon \in \mathbb{K}$. We notice that the new sharing $\mathbf{Z}'' = \mathbf{Z}' + (L'_2\epsilon, \epsilon, 0, \dots, 0)$ also represents the same unmasked value as $\mathbf{Z}'$ does. Indeed, $\langle L', Z' \rangle = X'$, and $\langle L', (L'_2\epsilon, \epsilon, 0, \dots, 0) \rangle = L'_2\epsilon + L'_2\epsilon = 0$. By choosing $\epsilon = \mathbf{Z}'_2 + \mathbf{Z}_2$, we get for $\mathbf{Z}''$:

$$\mathbf{Z}'' = (Z'_1 + L'_2(Z'_2 + Z_2), Z_2, Z'_3, \dots, Z'_n).$$

Therefore, $\mathbf{Z}''$ now has the same the second share (coordinate at position 2) with $\mathbf{Z}$. The complete homogenization is thus the repetition of this process for all the coordinates $i \in \{2, \dots, n\}$. Notice that this algorithm does leak information neither on $\mathbf{Z}$ nor on $\mathbf{Z}'$, since it consists only of additions of masks to a sharing from an independent sharing. It is akin to a refresh procedure albeit where the new masks are actually a compensation of $\mathbf{Z}'$ masks by those of $\mathbf{Z}$, while keeping the masking invariant of Eqn. 1. Actually, it is a refresh algorithm using the masks of the difference $\mathbf{Z} \oplus \mathbf{Z}'$. □

By using Algorithm 1, one can start with plaintext and key representation as Eqn. 8 and carry addition/multiplication (and refresh if needed) to implement any cryptographic algorithms like AES and end up with a ciphertext still with the form as Eqn. 8. Hence, verification can be done only at the very end. Another advantage of IPM-FD is its scalability, by choosing different values of k and n.

4 Security analysis of IPM-FD and optimal codes selection

The security level of IPM-FD can be characterized by three metrics, namely word-level security order d_w, bit-level security order d_b and number of detected faults d_f (for instance, if the number of faulted words is smaller than $d_f + 1$, then the fault will be detected). In this section, we show the security order of IPM-FD and how to choose optimal codes by interpreting IPM-FD as a coding problem.

4.1 Security of fault detection

We assess the security of IPM-FD against fault injection attacks in a coding theoretic approach. Assume a code of parameters $[n, k, d]_q$ over $\mathbb{F}_q$; there are three kinds of attackers in the state of the art:

- An attacker which can corrupt one to $d - 1$ symbols (elements of field $\mathbb{F}_q$). We assume here that faulting two symbols is somehow more difficult than faulting one symbol, etc. It is all the more difficult to fault, for the attacker, as more symbols must be corrupted simultaneously.
- An attacker which can randomly change a code word to a different one, which may not be a valid code word. We assume that the attacker has no control over the faulted value and if the faulted value is a valid code word, then the fault cannot be detected.
- An attacker which can choose the error ϵ that best suits him. In this scenario, the attacker will choose ϵ which maximizes her advantage, on replacing all code words z by $z + \epsilon$. This model assumes a much stronger attacker, but it does not always represent a realistic use case as the requirements (costs) are quite high. This model has been promoted initially by Mark Karpovsky et al. [18–20], who also proposed robust codes and algebraic manipulation detection (AMD) codes.

Accordingly, the probabilities to detect a fault in those three cases are:

- 100% for the first case when the number of faulted symbols $< d$. But this holds only if the verification can be done on each and every code word, which is not the case for us (we check only at the very end). Thus, we cannot claim any security level when chaining operations.
- $1 - 2^{k-n}$ for the second case. This detection rate is also valid end to end (i.e., with verification delayed on the ciphertext). Indeed, there are two cases: either the fault replaces a code word with a valid code word, and this will not be detected, neither by checking right on the targeted code word nor later on. Same reasoning otherwise: If the fault replaces a code word by a non-code word, then the non-code word will keep being a non-code word after all the operations (and we do not consider double faults here). Therefore, detection (in code or not) can be carried out at any point in time after the fault has been injected.
- $1 - |C \cap (C + \epsilon)|/|C|$ for the third case. Same reasoning as for the second case—this metric will stay unaltered throughout the computation.

In our IPM-FD setup, we support the last two models. Since we use the repetition code in IPM-FD, the minimum distance of the linear code C is $d_C = k$. Hence, the security in sense of fault injection attack is now assessed with respect to number of detected faults as:

$$d_f = k - 1. \tag{9}$$

It is obvious that any faults can be detected if the k copies of results are inconsistent.

4.2 Security order of IPM-FD on SCA resistance

The addition and refresh algorithms are secure since there is no degradation on masks; we focus on multiplication algorithm Algorithm 4, and we have the following Theorem 1.

Theorem 1 *The multiplication of IPM-FD in Algorithm 4 is $d_D^{\perp} - 1$-order secure.*

Proof The k times of IPMult multiplications at line 2 are secure at $(n - k)$th order [2]. After their application, the k shared variables $\mathbf{P}[j]$, $1 \le j \le k$, are masked by $N_{i,j}$ $(k + 1 \le i \le n, 1 \le j \le k)$ that are $(n - k) \times k$ uniformly distributed and i.i.d. random variables.

At step 6, indexed by i $(k + 1 \le i \le n)$, the contents of P_j are:

$$P_j = X_j X'_j + \left(\sum_{i'=k+1}^{i} L_{i',j} N_{i',1} \right) + \left(\sum_{i'=i+1}^{n} L_{i',j} N_{i',j} \right). \tag{10}$$

It is easy to see that any combinations of intermediate variations with mixed variables masked by $N_{i,j}$ and $N_{i,j'}$, for $j \ne j'$, require more intermediate values to be probed than strategies which focus on a given $N_{i,j}$ (for a given j).

The key-dependent variables which are only in $P_{i,1}$ (since homogenization process consists in turning $N_{i,j}$ into $N_{i,1}$) are those at:

- line 2: $X_1 X'_1 + \sum_{i=k+1}^{n} L_{i,1} N_{i,1}$, and the $(n - k)$ masks $N_{i,1}$ $(k + 1 \le i \le n)$;
- line 6: for $i = n$, $P_j = X_j X'_j + \sum_{i=k+1}^{n} L_{i,j} N_{i,1}$.

Finally, those shares are combined in an orderly manner as $\mathbf{P}$ (line 7). Together, they have the shape:

$$\mathbf{P} = (X_1, \ldots, X_k)\mathbf{G} + \mathbf{NH},$$

where $\mathbf{N} = (N_{k+1,1}, \ldots, N_{n,1}) \in \mathbb{K}^{n-k}$ is a uniformly distributed tuple of i.i.d random variables. Since $d_D^\perp - 1$ columns of $\mathbf{H}$ are independent [22, Theorem 10], which means if the attacker probes up to $d \leq (d_D^\perp - 1)$ variables, the secret X_j encoded as an element of $\mathbb{F}_{2^l}^{n-k+1}$ is perfectly masked. The security order of Algorithm 4 is $(d_D^\perp - 1)$. □

In summary, the security order at word-level d_w and bit-level d_b of IPM-FD corresponds to $(d_D^\perp - 1)$ at word level and $(d_D^{\perp\prime} - 1)$ bit level (by Code Expansion defined in Def. 2), respectively. In particular, the maximum word-level security order d_w is $(n-k)$, since $d_D^\perp \leq (n-k+1)$ from singleton bound [34], with equal if and only if $d_D^\perp$ is maximized.

4.3 Choosing optimal codes for IPM-FD

Two security orders d_w and d_b are connected to dual distance of D at word level and bit level, by encoding Eqn. 7 and Eqn. 8. Thus, we can search for minimal n satisfying the given requirements on the three parameters d_f, d_w and d_b. Since the best d_b is very difficult to obtain, we first search for codes given d_f and d_w and then find the best one with respect to optimal d_b. For the first step, Algorithm 5 is adopted for selecting codes with minimal n given d_f and d_w. In this algorithm, BKLC is short for "best known linear code."

Algorithm 5: Selecting codes given d_f and d_w.

```
input  : $l$ for $\mathbb{K} = \mathbb{F}_{2^l}$, $d_f$ for number of detected faults and $d_w$
         for word-level side-channel security
output: the minimal $n$ satisfying the requirements
1 $n \leftarrow d_w$ // $n$ is at least the minimum distance
  of the code generated by $\mathbf{H}^\perp$
2 while MinimumDistance([$BKLC(GF(2^l), n, d_f + 1)$] $< d_w$) [1]
  do
3     $n \leftarrow n + 1$
4 return $n$
```

The second step is to choose the best code with maximal bit-level security order d_b. We propose Algorithm 6 to select optimal codes with maximized d_b. Notice that this algorithm 6 is *conceptual*, as in line 3, it is not possible in practice to enumerate all codes. This line is to be understood according to either some algebraic code construction (parametric design pattern, greedy algorithm, etc.) or code random choice (using genetic algorithms, random generating matrices, etc.).

Algorithm 6: Choosing optimal codes with maximal d_b.

```
input  : $l$ for $\mathbb{K} = \mathbb{F}_{2^l}$, $d_f$ for number of detected faults, $d_w$ for
         word-level side-channel security and number of shares $n$
output: the maximal $d_b$ and optimal code $D$
1 $d_b \leftarrow d_w$      // Security order at bit-level is
  greater than word-level
2 $D_{opt} \leftarrow null$
3 forall the code $D = [n, d_f + 1, d_w + 1]_{2^l}$ do // Conceptual
4     $D_2 \leftarrow SubfieldRespresentation(D, GF(2))$
5     if $d_b <$ MinimumDistance($D_2$) then
6         $d_b \leftarrow MinimumDistance(D_2)$
7         $D_{opt} \leftarrow D$
8 return $d_b$, $D_{opt}$
```

We present some examples for codes in $\mathbb{F}_{2^8}$ in Table 2 (for $\mathbb{F}_{2^4}$ in Table 5, resp) calculated by Magma for small k and n. Interestingly, we compare the original IPM and IPM-FD with n and $n+1$ shares, respectively, since in IPM-FD redundancy is needed for fault detection. For IPM with $n = 3$, we have optimal parameters $d_w = 2$ and $d_b = 5$, while for IPM-FD with $n = 4$, $k = 2$, the optimal d_w and d_b are $d_w = 2$ and $d_b = 4$. Hence, there is a trade-off for fault detection, which sacrifices the bit-level side-channel resistance. For instance, for $k = 2$, we can detect one error.

We recall that the security order of IPM at bit level is given by the minimum distance of the code generated by $\mathbf{H}^\perp = (1, L_2, \ldots, L_n)$ (projected from $\mathbb{K} = \mathbb{F}_{2^l}$ to the binary ground field $\mathbb{F}_2^l$). Now, adding fault detection capability, the security order of IPM-FD becomes that of the minimum distance of the code generated by Eqn. 11. However, the

Table 2 Instances of codes with $X \in \mathbb{K} = \mathbb{F}_{2^8}$, d_b in IPM entries are consistent with results provided in [30]

	Inputs		Outputs of Algorithms 5 and 6		
	#faults d_f	d_w	n	d_b	Setting
IPM	0	0	1	0	$\mathbf{H}^\perp = (1)$
	0	1	2	3	$\mathbf{H}^\perp = (1\ \alpha^8)$
	0	2	3	7	$\mathbf{H}^\perp = (1\ \alpha^8\ \alpha^{26})$
	0	3	4	10^2	$\mathbf{H}^\perp = (1\ \alpha^8\ \alpha^{26}\ \alpha^{17})$
IPM-FD	1	0	2	0	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
	1	1	3	3	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & \alpha^8 \\ 0 & 1 & \alpha^{17} \end{pmatrix}$
	1	2	4	6	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & \alpha^8 & \alpha^{20} \\ 0 & 1 & \alpha^{27} & \alpha^7 \end{pmatrix}$

In Table 2, the maximal d_b for IPM codes with $n = 4$ shares in $\mathbb{F}_{2^8}$ is only 10 ($d_D^\perp = 11$), not 11 as shown in [30]

[1] BKLC is the short of the best known linear code in Magma [35].

minimum distance of this code is less than that generated by either $(1, L_{3,1}, L_{4,1}, \ldots, L_{n,1})$ or $(1, L_{3,2}, L_{4,2}, \ldots, L_{n,2})$.

$$\mathbf{H}'^{\perp} = \begin{pmatrix} 1 & 0 & L_{3,1} & L_{4,1} & \ldots & L_{n,1} \\ 0 & 1 & L_{3,2} & L_{4,2} & \ldots & L_{n,2} \end{pmatrix}. \quad (11)$$

4.4 Comparison between IPM-FD and Boolean masking with fault detection

We recall that, in the state of the art about masking countermeasures, Boolean masking (BM, [25, §4]) is presented as a particularly convenient masking scheme, since sharing and demasking only involve XOR operations. In contrast, IPM, in addition to field additions (XORs), is furthermore encumbered with field multiplication with constants (the $L_i \in \mathbb{K}$ values). This makes implementations more complex on programming (code size) and less efficient to implement. In practice, BM is thus a particular case of IPM, where all coefficients $L_i = 1 \in \mathbb{K}$.

Still, one historical advantage of IPM over BM, which initially justified for the scheme, is that, at a given side-channel security order at word level, IPM is more efficient at bit level (e.g., when the leakage model is the Hamming weight or the Hamming distance).

Now, in this paper, we put forward a second advantage of IPM, in the context of fault detection (FD). Table 3 compares IPM-FD with BM-FD in this respect. It clearly appears that fault detection is not straightforward in BM-FD, whereas it is for IPM-FD. As an example, when detecting one single fault ($d_f = 1$), and targeting a second-order protection in terms of word-level side-channel, IPM-FD manages to reach $d_w = 2$ with only $n = 4$ shares, thanks to:

$$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & \alpha^8 & \alpha^{20} \\ 0 & 1 & \alpha^{27} & \alpha^7 \end{pmatrix} \in \mathbb{F}_{2^8}^{2\times 4}.$$

However in Boolean masking scheme counterpart (i.e., in BM-FD), it is not possible to reach a minimum distance for $H^{\perp}$ of value $= 3$ with a code length $n = 4$. Indeed, in systematic form, it would look as:

$$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & ? & ? \\ 0 & 1 & ? & ? \end{pmatrix}.$$

Now, as the minimum distance is 3, the weight of each line must be 3. Therefore, all $2 + 2$ question marks ("?" symbol) must be nonzero, that is, equal to 1 (in the case of BM). Hence, the difference between the two lines is equal to $\begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}$, which has a weight $= 2$, therefore a contradiction. However, let us notice that the problem can be solved by considering a length extended by one, that is:

$$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & ? & ? & ? \\ 0 & 1 & ? & ? & ? \end{pmatrix},$$

Table 3 Comparison of d_w, d_b between IPM-FD and BM-FD (Boolean masking with fault detection) for $X \in \mathbb{K} = \mathbb{F}_{2^8}$, and for $d_w \in \{1, 2, 3\}$. Note that here we set $d_f = 1$ (meaning $k = 2$) for a fair comparison

d_w	IPM-FD		BM-FD	
	n	d_b	n	d_b
0	2	0	2	0
1	3	**3**	3	**1**
2	**4**	**6**	**5**	**2**

where among the three question marks in one line, at least two are nonzero ($= 1$). Knowing that the constraint is not only to have the number of ones ≥ 3 in each line, but also in the sum of the two lines, we can use:

$$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \in \mathbb{F}_{2^8}^{2\times 5}.$$

But its length is $n = 5$, i.e., larger by one unit compared to IPM case, where constants can be chosen arbitrarily in the whole $\mathbb{F}_{256}$ and not only in $\{0, 1\} \subset \mathbb{F}_{256}$.

Summarizing up, as shown in Table 3, the IPM-FD is better than BM-FD in two aspects given the same d_f. Firstly, IPM-FD needs less shares than BM-FD when achieving the same word-level security order (denoted in **red bold font** in Table 3). Secondly, the bit-level security order in IPM-FD is much higher than in BM-FD given the same d_w (denoted in **black bold font** in Table 3). It is worthy noting that the advantages of IPM-FD over BM-FD become much larger when the number of shares increases. However, in order to find the good or even optimal codes for IPM-FD, it is necessary to turn to DSM scheme.

5 Practical implementation and performances

We implement IPM-FD scheme on AES-128 based on (thanks to) open-source implementation of masked AES by Coron *et al.* [12,13]. All the computations are made with additions, multiplications and lookups in some pre-computed tables. The random number generator comes from the https://libsodium.gitbook.io/ Sodium library [16]. Each sensitive variable ($16 \times (10+1)$ subkeys from the *Key Schedule* routine and 16 bytes in state array) is masked into n shares using $n-k$ random bytes. In particular, regarding nonlinear operations, the S-box of a masked value is computed online instead of the 256-sized table, where its polynomial expression obtained via Lagrange interpolation is:

$$x \in \mathbb{F}_{2^8} \mapsto \texttt{63} + \texttt{8f}x^{127} + \texttt{b5}x^{191} + \texttt{01}x^{223} + \texttt{f4}x^{239} + \texttt{25}x^{247} + \texttt{f9}x^{251} + \texttt{09}x^{253} + \texttt{05}x^{254}.$$

After demasking a shared variable, we check that the data have no faults injected by comparing the k copies and raising

Table 4 Performance comparison of IPM-FD with and without fault detection. Speed is the runtime in milliseconds averaged over 1000 runs on a PC with 2.8 GHz 6-core processor, and *random* is the number of generated random bytes when masking and refreshing

Security order	IPM (baseline)	Two consecutive executions of IPM	IPM-FD $k = 2$
$d_w = 1$	$n = 2$ $(d_b = 3)$, $speed = 1.52$, $random = 1936$	$n = 2$ $(d_b = 3)$, $speed = 3.04$, $random = 3872$	$n = 3$ $(d_b = 3)$, $speed = 2.93$, $random = 3856$
$d_w = 2$	$n = 3$ $(d_b = 7)$, $speed = 2.25$, $random = 5152$	$n = 3$ $(d_b = 7)$, $speed = 4.50$, $random = 10304$	$n = 4$ $(d_b = 6)$, $speed = 4.31$, $random = 10272$

an alarm if any fault is detected. Our implementation works for any $n \geq k$. Specially, for $n < 5$ and $k < 3$ we choose the best known linear code (BKLC) D obtained with `Magma`; otherwise, we randomly generate a matrix for masking.

Our implementation of IPM-FD on AES (in `C`) is publicly available [9]. Furthermore, the optimal linear codes for IPM by an exhaustive study are available [10].

5.1 Performance evaluation

We make a comparison for the same security order at word level, between:

- No fault detection (classic IPM, $k = 1$)—this is our reference
- Single fault detection by temporal redundancy (repeat twice the IPM computation)
- Single fault detection embedded into IPM (so-called IPM-FD for $k = 2$)

Performance-wise, Table 4 shows that two fault detection strategies (temporal repetition and IPM-FD) are at essentially the same cost.

But if we consider the most time-consuming operation—the field multiplication: the number of field multiplications in IPM on n shares (Algorithm 5 of [2]) is $3n^2 - n$. However, the number of multiplications in IPM-FD on n shares is:

- $k(3(n-k+1)^2 - (n-k+1))$ regarding the k IPM multiplications on $n-k+1$ shares,
- $(k-1)(n-k)$ regarding the $(k-1)$ homogenizations.

Hence, a total complexity of $k(3(n-k+1)^2 - (n-k+1)) + (k-1)(n-k)$ is:

- $3n^2 - n$ for IPM-FD with $k = 1$,
- $6n^2 - 13n + 6$ for IPM-FD with $k = 2$.

Now, we have that $2 \times (3n^2 - n) > 6n^2 - 13n + 6$, which are shown in Fig. 5. Therefore, it is more interesting, complexity-wise, to use IPM-FD for $k = 2$ than repeating a computation twice.

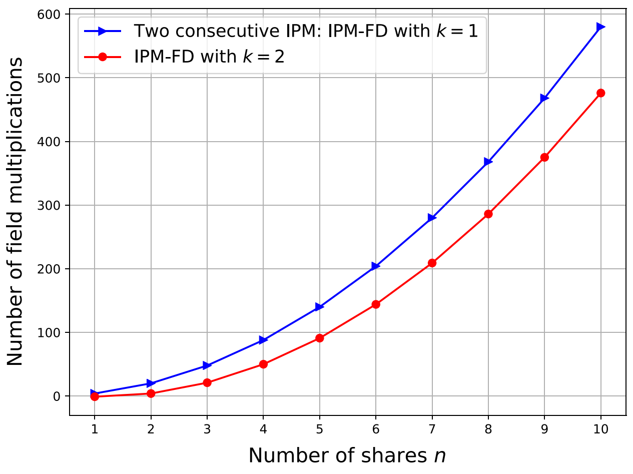

Fig. 5 Comparison of number of field multiplications in terms of n, where $k = 1$ for IPM and $k = 2$ for IPM-FD, respectively

Notice that temporal redundancy is prone to fault injection attacks [29,32], whereby an attacker would reproduce exactly the same fault on the repeated executions. Therefore, our IPM-FD is intrinsically stronger against fault attacks, at the same cost in terms of execution speed.

6 Conclusion and perspectives

IPM shows an advantageous property—higher security order at bit level d_b than at word level—as a promising alternative to Boolean masking. In this paper, we propose a novel end-to-end fault detection scheme called IPM-FD, which is a IPM-like scheme to detect faults by redundancy on secrets rather than on masks. The IPM-FD is also a unified scheme to resist side-channel analysis and fault injection attack simultaneously. We also present an example by applying IPM-FD to AES and provide a comparison on performance with different settings.

As a perspective, we notice that the performances of both IPM and IPM-FD can be improved by choosing small (or sparse) values for $L_{i,j} \in \mathbb{K}$ scalars. This strategy is similar to that already employed by Rijndael inventors, for instance when designing the MixColumns operation. This raises the question of finding codes with sparse matrices of high dual distance.

Secondly, we show in Tables 2, 5 and 6 for results by an exhaustive study, which is very time-consuming and even impossible to find the optimal one when the number of shares n gets larger. Hence, a systematic (e.g., algebraic) construction of better codes than mere repetition codes is much more preferable and could be leveraged. However, it is still an open problem to construct optimal or suboptimal codes for IPM-FD. One possible approach is to convert some constructions [6] in DSM to IPM-FD which needs further study.

Besides, we notice that our fault detection paradigm applies also to the case of Boolean masking, i.e., IPM, where all constants $L_{i,j}$ are equal to 1, which can also enable enhancements of currently deployed software code with respect to detection of perturbations.

Acknowledgements This work has been partly financed via the project TEAMPLAY (https://teamplay-h2020.eu/), a project from European Union's Horizon2020 research and innovation program, under grant agreement No. 779882, and also supported by SECODE project (https://secode.telecom-paristech.fr/) under grant No. ANR-15-CHR2-0007 funded by the CHIST-ERA program and coordinated by ANR.

An optimal codes for IPM-FD with $k = 2$

By using Magma [35], we present some instances for IPM-FD with $k = 2$, in particular $\mathbb{K} = \mathbb{F}_{2^4}$ in Table 5 and $\mathbb{K} = \mathbb{F}_2$ in Table 6, respectively. Interestingly, we notice that for $\mathbb{K} = \mathbb{F}_2$ the best minimum distance of $\mathbf{H}^{\perp}$ is equal to *BKLC(GF(2), n, 2)*, where n is the same as in Table 6.

Table 5 Examples with $\mathbb{K} = \mathbb{F}_{2^4}$, d_b and d_w are side-channel security orders at bit level and word level, respectively

	Inputs		Outputs of Algorithms 5 and 6		
	#faults d_f	d_w	n	d_b	Setting
IPM	0	0	1	0	$\mathbf{H}^{\perp} = (1)$
	0	1	2	2	$\mathbf{H}^{\perp} = (1\ \alpha^5)$
	0	2	3	5	$\mathbf{H}^{\perp} = (1\ \alpha^5\ \alpha^{10})$
	0	3	4	7	$\mathbf{H}^{\perp} = (1\ \alpha^5\ \alpha^9\ \alpha^{13})$
	0	4	5	9	$\mathbf{H}^{\perp} = (1\ \alpha^5\ \alpha^9\ \alpha^{12}\ \alpha^1)$
	0	5	6	11	$BKLC(GF(2), 4 * 6, 4) \simeq [24, 4, 12]$
IPM-FD	1	0	2	0	$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
	1	1	3	2	$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & \alpha^5 \\ 0 & 1 & \alpha^{10} \end{pmatrix}$
	1	2	4	4	$\mathbf{H}^{\perp} = \begin{pmatrix} 1 & 0 & \alpha^5 & \alpha^{11} \\ 0 & 1 & \alpha^{11} & \alpha^4 \end{pmatrix}$

Table 6 Examples with $\mathbb{K} = \mathbb{F}_2$, d_w and d_b are security orders at word level and bit level, respectively. In this case, the same codes can also be used in BM-FD, while BM-FD is defined over $\mathbb{K} = \mathbb{F}_{2^l}$

	Inputs		Outputs of Algorithm 5 and Algorithm 6		
	#faults d_f	d_w	n	d_b	Setting
IPM	0	0	1	0	$\mathbf{H}^\perp = (1)$
	0	1	2	1	$\mathbf{H}^\perp = (1\ 1)$
	0	2	3	2	$\mathbf{H}^\perp = (1\ 1\ 1)$
	0	3	4	3	$\mathbf{H}^\perp = (1\ 1\ 1\ 1)$
	0	4	5	4	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1)$
	0	5	6	5	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1\ 1)$
	0	6	7	6	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1\ 1\ 1)$
	0	7	8	7	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$
	0	8	9	8	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$
	0	9	10	9	$\mathbf{H}^\perp = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$
IPM-FD (BM-FD)	1	0	2	0	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
	1	1	3	1	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$
	1	2	5	2	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$
	1	3	6	3	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$
	1	4	8	4	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$
	1	5	9	5	$\mathbf{H}^\perp = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$

References

1. Ali, Subidh, Mukhopadhyay, Debdeep, Tunstall, Michael: Differential fault analysis of AES: towards reaching its limits. J. Cryptogr. Eng. **3**(2), 73–97 (2013)
2. Balasch, J., Faust, S., Gierlichs, B.: Inner product masking revisited. In: Elisabeth, O., Marc F., editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pp 486–510. Springer, (2015)
3. Balasch, J., Faust, S., Gierlichs, B., Paglialonga, C., Standaert, F.-X.: Consolidating inner product masking. In: Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pp. 724–754. Springer, (2017)
4. Barthe, Gilles, Belaïd, Sonia, Dupressoir, François, Fouque, Pierre-Alain, Grégoire, Benjamin: Compositional verification of higher-order masking: application to a verifying masking compiler. IACR Cryptol. ePrint Arch. **2015**, 506 (2015)
5. Bringer, J., Carlet, C., Chabanne, H., Guilley, S., Maghrebi, H.: Orthogonal direct sum masking—a smartcard friendly computation paradigm in a code, with builtin protection against side-channel and fault attacks. In: David Naccache and Damien Sauveron, editors, *Information Security Theory and Practice. Securing the Internet of Things - 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, volume 8501 of *Lecture Notes in Computer Science*, pp. 40–56. Springer, (2014)
6. Carlet, C., Güneri, C., Mesnager, S., Özbudak, F.: Construction of some codes suitable for both side channel and fault injection attacks. In: Lilya Budaghyan and Francisco Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields—7th International Workshop, WAIFI 2018, Bergen, Norway, June 14-16, 2018, Revised Selected Papers*, volume 11321 of *Lecture Notes in Computer Science*, pp. 95–107. Springer, (2018)
7. Chakraborty, Abhishek, Mazumdar, Bodhisatwa, Mukhopadhyay, Debdeep: A combined power and fault analysis attack on protected grain family of stream ciphers. IEEE Trans. CAD Integr. Circuits Syst. **36**(12), 1968–1977 (2017)
8. Cheng, W., Carlet, C., Goli, K., Danger, J.-L., Guilley, S.: Detecting faults in inner product masking scheme—IPM-FD: IPM with fault detection, August 24 2019. In: 8th International Workshop on Security Proofs for Embedded Systems (PROOFS). Atlanta, GA, USA (2019)
9. Cheng, W., Carlet, C., Goli, K., Danger, J.-L., Guilley, S.: Detecting faults in inner product masking scheme—IPM-FD: IPM with fault detection, August (2019). https://github.com/Qomo-CHENG/IPM-FD
10. Cheng, W., Guilley, S., Danger, J.-L., Carlet, C., Mesnager, S.: Optimal Linear Codes for IPM, January (2020). https://github.com/Qomo-CHENG/OC-IPM
11. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M.: Passive and active combined attacks on AES. In: *FDTC*, pp. 10–18. IEEE Computer Society, 21 August 2010. Santa Barbara, CA, USA. (2010) https://doi.org/10.1109/FDTC.2010.17
12. Coron, J.-S.: HTable countermeasure against side-channel attacks—reference implementation for the masking scheme presented in [13]. https://github.com/coron/htable
13. Coron, J.-S.: Higher order masking of look-up tables. In: Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume

8441 of *Lecture Notes in Computer Science*, pp. 441–458. Springer (2014)

14. Coron, J.-S., Prouff, E., Rivain, M.: Side channel cryptanalysis of a higher order masking scheme. In: Pascal, P., Ingrid, V., editors, *CHES*, volume 4727 of *LNCS*, pp. 28–44. Springer (2007)
15. Danger, Jean-Luc, Guilley, Sylvain, Heuser, Annelie, Legay, Axel, Tang, Ming: Physical security versus masking schemes. In: Koç, Çetin Kaya (ed.) Cyber-Physical Systems Security, pp. 269–284. Springer, Berlin (2018)
16. Denis, F.: The Sodium cryptography library, Jul (2019)
17. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pp. 463–481. Springer, August 17–21 2003. Santa Barbara, California, USA (2003)
18. Karpovsky, M.G., Kulikowski, K.J., Wang, Z.: Robust error detection in communication and computation channels. In: *in Proceedings of International Workshop on Spectral Techniques* (2007)
19. Karpovsky, Mark G., Nagvajara, Prawat: Optimal codes for minimax criterion on error detection. IEEE Trans. Inf. Theory **35**(6), 1299–1305 (1989)
20. Karpovsky, Mark G., Taubin, Alexander: New class of nonlinear systematic error detecting codes. IEEE Trans. Inf. Theory **50**(8), 1818–1820 (2004)
21. Kirschbaum, M., Popp, T.: Evaluation of a DPA-resistant prototype chip. In: *ACSAC*, pp. 43–50. IEEE Computer Society, 7-11 December 2009. Honolulu, Hawaii (2009)
22. Jessie MacWilliams, F., Sloane, Neil J .A.: The Theory of Error-Correcting Codes. Elsevier, Amsterdam, North Holland (1977). ISBN: 978-0-444-85193-2
23. MacWilliams, F.J., Sloane, N.J.A. Neil James A.: *The theory of error correcting codes*. North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977. Includes index (1977)
24. Mangard, S., Oswald, E., Popp, T.: *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, http://www.dpabook.org/ (2006)
25. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pp. 150–164. Springer (2000)
26. Monteiro, C., Takahashi, Y., Sekine, T.: Low power secure AES S-box using adiabatic logic circuit. In: *2013 IEEE Faible Tension Faible Consommation*, pp. 1–4, June (2013)
27. Moore, Simon, Anderson, Ross, Mullins, Robert, Taylor, George, Fournier, Jacques JA: Balanced self-checking asynchronous logic for smart card applications. J. Microprocess. Microsyst. **27**(9), 421–430 (2003)
28. Ngo, X.T., Bhasin, S., Danger, J.-L., Guilley, S., Najm, Z.: Linear complementary dual code improvement to strengthen encoded circuit against hardware Trojan horses. In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pp. 82–87. IEEE (2015)
29. Patranabis, S., Chakraborty, A., Nguyen, P.H., Mukhopadhyay, D.: A biased fault attack on the time redundancy countermeasure for AES. In: Stefan, M., Axel, Y.P., editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pp. 189–203. Springer (2015)
30. Poussier, R., Guo, Q., Standaert, F.-X., Carlet, C., Guilley, S.: Connecting and improving direct sum masking and inner product masking. In: Thomas, E., Yannick, T., editors, *Smart Card Research and Advanced Applications—16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pp. 123–141. Springer (2017)
31. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Stefan, M., François-Xavier, S., editors, *CHES*, volume 6225 of *LNCS*, pp. 413–427. Springer (2010)
32. Saha, S., Jap, D., Breier, J., Bhasin, S., Mukhopadhyay, D., Dasgupta, P.: Breaking redundancy-based countermeasures with random faults and power side channel. In: *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018, Amsterdam, The Netherlands, September 13, 2018*, pp. 15–22. IEEE Computer Society (2018)
33. Schneider, T., Moradi, A., Güneysu, T.: Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pp. 302–332. Springer (2016)
34. Singleton, Richard C.: Maximum distance q -nary codes. IEEE Trans. Inf. Theory **10**(2), 116–118 (1964)
35. University of Sydney (Australia). Magma Computational Algebra System. http://magma.maths.usyd.edu.au/magma/, Accessed on Aug 22, 2014
36. Wang, W., Standaert, F.-X., Yu, Y., Pu, S., Liu, J., Guo, Z., Gu, D.: Inner product masking for bitslice ciphers and security order amplification for linear leakages. In: Kerstin, L.-R., Michael, T., editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pp. 174–191. Springer (2016)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.